

09/769,953

L Number	Hits	Search Text	DB	Time stamp
1	12	((("5857086") or ("5867645") or ("5889970") or ("5892964") or ("5923860") or ("6018810"))).PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 16:43
2	14	((("5361267") or ("5701409") or ("5724528") or ("5764924") or ("5781918") or ("5867645") or ("5884027"))).PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:09
3	3983	(accelerat\$4 adj graphic\$ adj port) or AGP	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:11
4	346129	(peripheral adj component ad interconnect) or PCI	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:12
5	1976	((accelerat\$4 adj graphic\$ adj port) or AGP) with ((peripheral adj component ad interconnect) or PCI)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:12
6	3835044	detect\$ or monitor\$4 or track\$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:13
7	5084838	error\$ or fault\$4 or problem\$ or fail\$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:14
8	174525	(detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:15
9	9999	(first or primary or master) adj bus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:16
10	46	((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:16

11	0	((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) with (((accelerat\$4 adj graphic\$ adj port) or AGP) with ((peripheral adj component ad interconnect) or PCI))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:17
12	1	((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and (((accelerat\$4 adj graphic\$ adj port) or AGP) with ((peripheral adj component ad interconnect) or PCI))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:17
13	1	((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and ((accelerat\$4 adj graphic\$ adj port) or AGP) and ((peripheral adj component ad interconnect) or PCI)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:18
14	4005	(714/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:18
15	3093	(710/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:18
16	677	(712/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:19
17	7694	((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:19
18	12	((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and (((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:31
19	6547	parity adj error\$	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:31
20	15	((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and (parity adj error\$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:31

21	12533	(input/output or i/o)adj bus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:32
22	4	(((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and ((input/output or i/o)adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:35
23	1	(((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and (parity adj error\$)) and ((input/output or i/o)adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:35
24	10	(((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and (parity adj error\$)) not (((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and (((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.))) or (((detect\$ or monitor\$4 or track\$4) adj3 (error\$ or fault\$4 or problem\$ or fail\$4)) with ((first or primary or master) adj bus)) and ((input/output or i/o)adj bus)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:35

us

register through said bus without synchronizing said read operation
with said

CPU of said respective supervised appa

US-PAT-NO: 5717852

DOCUMENT-IDENTIFIER: US 5717852 A

TITLE: Multiple bus control method and a system thereof

----- KWIC -----

Abstract Text - ABTX (1):

In the bus constitution in an information processing unit for transferring data between a bus-master and bus-slave via a plurality of buses, if a fault occurs in a bus when data is transferred between the bus-master and bus-slave via the buses and the H side fault detection/reporting circuit of the bus-slave or the H side fault detection/processing circuit of the bus-master detects the fault, the data transfer via the bus is stopped and information indicating that the bus is faulty is stored in the bus status information keeping circuit. When the data transfer via the bus ends normally thereafter, the bus selector confirms that the bus is normal from the information of the bus status information keeping circuit and activates the L side bus controller so as to operate the bus and to transfer the data which cannot be transferred due to the bus fault via the bus.

Brief Summary Text - BSTX (17):

More specifically according to embodiments which will be described later, the present invention is a multiple bus control method in a system for transferring data between the bus-master and bus-slave by occupying a plurality of buses, which detects whether there is a fault in each bus for transferring data, stops the data transfer by the failed bus, and retransfers the data via a fault-free bus.

Brief Summary Text - BSTX (18):

The system of the present invention is a system for transferring data between the bus-master and bus-slave by occupying a plurality of buses. The bus-slave includes bus control means for controlling access from the bus-master for each bus and fault detection/reporting means for detecting whether

there is
a fault in each bus when transferring data and for reporting the
detection
result to the bus-master. The bus-master includes fault
detection/processing
means for detecting whether there is a fault in each bus when
transferring data
and for generating information indicating whether or not the buses can
be used
according to the detection result and the report from the fault
detection/reporting means of the bus-slave, information keeping means
for
keeping the information generated by the fault detection/processing
means, bus
selection means for confirming whether or not the buses can be used
according
to the information kept by the information keeping means when
transferring
data, and bus control means for driving the usable bus according to the
confirmation of the bus selection means when data transfer starts and
for
stopping the data transfer by the unusable bus which is confirmed by
the bus
selection means after the data transfer starts. The bus selection
means
selects and drives the usable bus by starting the bus control means
after data
transfer and retransfers the data, which is transferred halfway via the
unusable bus, via the bus which is selected and driven.

Brief Summary Text - BSTX (22):

When one of the buses fails during the data transfer, in the case of
data
transfer from the bus-master to the bus-slave, the fault is detected by
the
fault detection/reporting means of the bus-slave and reported to the
fault
detection/processing means of the bus-master. In the case of data
transfer
from the bus-slave to the bus-master, the fault is detected by the
fault
detection/processing means of the bus-master and the fault
detection/processing
means updates the information of the information keeping means
according to the
fault. The bus selection means confirms the failed bus from this
updating of
the information of the information keeping means and stops the data
transfer by
the bus. When the current data transfer ends, the bus selection means
confirms
the usable bus from the information of the information keeping means
and
retransfers the data which is stopped in transfer via this bus.

Detailed Description Text - DETX (30):

Even if the H side bus controller 5 or the L side bus controller 7 of the bus-master 2A or 2B fails or the H side bus controller 10 or the L side bus controller 12 of the bus-slave 9 fails, the H side fault detection/processing circuit 6 or the L side fault detection/processing circuit 8, or the H side fault detection/reporting circuit 11 or the L side fault detection/reporting circuit 13 detects the fault in the same way as with a case that the H side bus 14 or the L side bus 15 fails, and data is not transferred incorrectly, and since all the buses are occupied by the bus-master and bus-slave which transfer data, no data is transferred between another bus-master and bus-slave and no different data is transferred at the same time.

Current US Cross Reference Classification - CCXR (1):

714/2

US-PAT-NO: 5680537

DOCUMENT-IDENTIFIER: US 5680537 A

TITLE: Method and apparatus for isolating an error
within a computer system that transfers data via an
interface device

----- KWIC -----

Abstract Text - ABTX (1):

A method and apparatus for isolating an error in a system having a controller or the like which access a user via an interface device. The controller or the like may be coupled to the interface device via a first bus and the interface device may be coupled to the user via a second bus. The controller or the like may detect an error in a data transfer from the user to the controller via the interface device, and may isolate the error to the second bus/interface device or the first bus/controller. This up-front error isolation may reduce the amount of analysis required by a service technician after a corresponding PC board or the like is removed from the system, thereby reducing the cost thereof.

Brief Summary Text - BSTX (17):

The present invention overcomes many of the disadvantages of the prior art by providing a method and apparatus for isolating an error within a computer system which transfers data via an interface device. That is, the present invention may provide a means for isolating an error in a system having a controller or the like which access a user via an interface device. The controller may be coupled to the interface device via a first bus and the interface device may be coupled to the user via a second bus. The controller may detect an error in a data transfer from the user to the controller via the interface device, and may isolate the error to the second bus/interface device or the first bus/controller. This up-front error isolation may reduce the

amount of analysis required by a service technician after a corresponding PC board or the like is removed from the system, thereby reducing the cost thereof.

Brief Summary Text - BSTX (19):

The controller may receive data transfers from the user having the incompatible interface via the interface device. The controller may have an error detection capability for detecting errors received thereby. The controller may thus detect an error provided by the interface device. To isolate the source of the detected error, the controller may read a status word from the status register of the interface device via the first bus. It is contemplated that the controller may ignore or otherwise prevent corrupted data from passing through the controller while performing error isolation. The controller may then perform error detection on the status word. If an error is detected, it is assumed that the source of the error is the first bus or the controller itself. If an error is not detected, the error bit of the status word may be analyzed. If the error bit indicates that the interface device detected an error on the first bus, it is assumed that the source of the error is the second bus or the interface device. Although this invention does not completely isolate the source of the error, the information provided thereby may greatly reduce the amount of analysis required by a service technician and/or a dedicated test system.

Detailed Description Text - DETX (2):

FIG. 1 is a block diagram of a first exemplary embodiment of the present invention. The block diagram is generally shown at 10. The exemplary embodiment may provide a means for isolating an error in a system having a controller 12 or the like which access a user 20 via an interface device 16. Controller 12 or the like may be coupled to interface device 16 via a first bus 18 and interface device 16 may be coupled to user 20 via a second bus 22. Controller 12 or the like may detect an error in a data transfer from user 20 to controller 12 via an error detection capability 28, and may isolate the error to the second bus 22 and/or interface device 16 or the first bus 18

and/or controller 12. This up-front error isolation may reduce the amount of analysis required by a service technician after a corresponding PC board or the like is removed from the system, thereby reducing the cost thereof.

Detailed Description Text - DETX (4):

Controller 12 may receive data transfers from user 20 via interface device 16 as described above. Controller 12 may have error detection capability 28 for detecting errors received thereby. Controller 12 may thus detect an error provided by interface device 24. To isolate the source of the detected error, an error control block 30 within controller 12 may initiate a read operation of the status register 26 of interface device 16 via first bus 18, thereby resulting in a status word. It is contemplated that controller 12 may ignore or otherwise prevent corrupted data provided by interface device 16 from passing through or into controller 12 while performing error isolation.

Controller 12 may then perform error detection on the status word via the error detection capability 28. If an error is detected, it is assumed that the source of the error is first bus 18 or controller 12. If an error is not detected, the error bit of the status word may be analyzed. If the error bit indicates that interface device 16 detected an error on first bus 22, it is assumed that the source of the error is second bus 22 or interface device 16. Although this invention does not completely isolate the source of the error, the information provided thereby may greatly reduce the amount of analysis required by a service technician and/or a dedicated test system.

Claims Text - CLTX (9):

i. first bus error detection means for performing error detection on said number of data elements as said number of data elements are read over said first bus;

Claims Text - CLTX (10):

ii. reading means coupled to said storing means for reading said error code from said storing means when said first bus error detection means detects an error;

Claims Text - CLTX (11):

iii. error code error detection means coupled to said reading means for performing error detection on said error code thereby resulting in a slave read error code when said first bus error detection means detects an error; and

Claims Text - CLTX (24):

i. first bus error detection means for performing error detection on said number of data elements as said number of data elements are read over said first bus;

Claims Text - CLTX (25):

ii. reading means coupled to said storing means for reading said error code from said storing means when said first bus error detection means detects an error;

Claims Text - CLTX (42):

i. a first bus error detection circuit for performing error detection on said number of data elements as said number of data elements are read over said first bus;

Claims Text - CLTX (43):

ii. a reading circuit coupled to said storing circuit for reading said error code from said storing circuit when said first bus error detection circuit detects an error;

Claims Text - CLTX (46):

13. A method for isolating an error within a computer system wherein the computer system has a bus user, an interface device, and a controller, wherein the controller is coupled to said interface device via a first bus and the interface device is coupled to the bus user via a second bus, the interface device having an error detection capability wherein the error detection capability sets an error code if an error is detected on said second bus, the method comprising the steps of:

Claims Text - CLTX (58):

14. A method for isolating an error within a computer system wherein the computer system has a bus user, an interface device, and a controller, wherein the controller is coupled to said interface device via a first bus and the interface device is coupled to the bus user via a second bus, the interface device having an error detection capability wherein the error detection capability sets an error code if an error is detected on said second bus, the interface device providing a number of data elements to the controller, the method comprising the steps of:

Claims Text - CLTX (87):

16. A method for isolating an error within a computer system wherein the computer system has a disk storage element, a disk controller, and a DSDC, Data Save Disk Chip, element, wherein the DSDC, Data Save Disk Chip, element is coupled to said disk controller via a first bus and the disk controller is coupled to the disk storage element via a second bus, the disk controller having an error detection capability wherein the error detection capability sets an error code if an error is detected on said second bus, the method comprising the steps of:

Current US Original Classification - CCOR (1):

714/5

US-PAT-NO: 4792950
DOCUMENT-IDENTIFIER: US 4792950 A
TITLE: Multiplex wiring system

----- KWIC -----

Brief Summary Text - BSTX (11):

In one aspect of the invention, the object and advantages described above and others are obtained by a multiplex wiring system comprising: transmitting means for transmitting data on a first bus and for transmitting complementary data on a second bus, the complementary data being the complement of the data transmitted on the first bus; first receiver means coupled to the first bus; second receiver means coupled to the second bus; differential receiver means coupled to both buses for combining the data and the complementary data thereby achieving substantial noise immunity; detector means for detecting a predetermined data sequence, preferably a start-of-message, in each of the receiver means; decoder means coupled to the detector means for providing an indication of a fault, preferably including the type of fault and the location of the fault; and data selector means responsive to the decoder means for selecting data from one of the receivers based upon the fault indication.

Current US Original Classification - CCOR (1):
714/4

TDB-ACC-NO: NN890656

DISCLOSURE TITLE: SPD I/O Bus Interactive Test System

PUBLICATION-DATA: IBM Technical Disclosure Bulletin, June 1989, US

VOLUME NUMBER: 32

ISSUE NUMBER: 1

PAGE NUMBER: 56 - 59

PUBLICATION-DATE: June 1, 1989 (19890601)

CROSS REFERENCE: 0018-8689-32-1-56

DISCLOSURE TEXT:

- A controlled, repeatable means of initiating a wide variety of bus operations, detecting and injecting bus errors, and monitoring activity on an IBM SPD I/O bus is provided in the following.

- The SPD bus interactive test system (BITS) runs on the SPD I/O

Bus Test Vehicle (BTV) hardware (Fig. 1). The BTV consists of a personal computer (PC), a TESLA I/O Processor (IOP) card, and an associated error injection/error detection (EI/ED) I/O adapter

(IOA)

card. The PC communicates to the TESLA IOP via a general-purpose interface bus (GPIB) IEEE-488|. ***** SEE ORIGINAL DOCUMENT

The TESLA IOP controls the EI/ED card and accepts status from it via the IOA bus. The TESLA IOP contains a microprocessor, memory,

and an interface that allows it to communicate over the SPD I/O

bus.

BITS microcode runs inside the TESLA IOP, and can also access registers inside the EI/ED card.

- The EI/ED card contains circuitry that allows it to detect errors and inject errors on the SPD I/O bus. It can accept commands

and parameters from the TESLA IOP. It can issue an interrupt to the

TESLA when error detection or injection status is available.

- The TESLA portion of BITS is part of the BTV RAM Code. The PC

portion of BITS is written in C and compiled with the IBM C Compiler.

BITS software is part of the Operator Console Program (OCP). The IBM PC GPIB Adapter Programming Support Routines is used in the PC portion of BITS to communicate via the GPIB bus. Full Screen Executive (FSX/PC) is used to implement PC screens. The TESLA portion of BITS is written in PL.8, and compiled with the PL.8 Compiler. The TESLA portion uses the BTV ROS Code. The TESLA portion also uses the X-Series Control Program (XCP), which is part of the IBM 6030 Communications Adapter microcode.

- BITS consists of two separate programs that work together.

The PC portion runs on the IBM PC, PC/XT, PC/AT, PS/2, or compatible PCs under the DOS operating system. The PC portion of BITS is a functional component of OCP, a program that contains the user interface for the SPD I/O BTV. The TESLA portion runs in the TESLA IOP under the XCP operating system. BITS software structure is shown in Fig. 2.

- The TESLA IOP portion of BITS is a functional component of the BTV RAM code. The BTV RAM code, together with the BTV ROS code, issues and accepts SPD I/O Bus operations. The BTV RAM code also controls and services the EI/ED hardware. Finally, the BTV RAM code accepts and responds to commands from the PC via the GPIB interface.

- BITS alleviates the need for writing special programs to perform SPD I/O Bus level testing from an IOP. Instead, the BITS Operation List Definition screen allows the user to create a customized list of virtually any sequence of bus operations. This operation list can be saved in a file for future use.

- The operation list can be as simple as a single unit message, or can be a mixture of up to 10 different operations, each with its own repetition count. The entire list can be repeated up to 65,535 times or made to run indefinitely. The BITS Operation List can include the following: 1) sending unit messages, 2) waiting for unit messages, 3) initiating storage operations, 4) performing error detection, 5) performing error injection.

- The BITS Operation List Definition screen also allows the user to input parameters that initialize the test so that: The test stops (or continues) when the first bus error is detected, only error status (or all status) is reported during testing and status received during the test is logged (or is not logged) to a file.

- BITS screens allow the user to define both unit operations and storage operations. Illegal and invalid operations can be defined, allowing greater error recovery test coverage.

- The BITS Unit Operation Definition screen allows the user to define up to 10 unit operations. The following attributes of each unit operation can be explicitly defined in this session: slave bus unit address, bus command, bus priority, select cycle word and data cycle words.

- Similarly, the BITS Storage Operation Definition screen allows the user to define up to 10 storage operations and control the following attributes: data transfer direction, slave bus unit address, bus priority, length of data transfer, line length, data storage address in the BTV and data storage address in the host processor.

- Additionally, BITS provides a storage write/read/compare

option

that is controlled from this screen. This option allows the user to

transfer a block of data to the host, transfer it back, and compare the received data to the original. If a mismatch occurs, the user will be informed.

- The BITS Storage Operation Definition screen provides three methods for initializing data that is to be transferred from storage

inside the TESLA IOP: 1) Transfer from storage as is 2) Initialize storage from the 64-bit pattern provided by the user 3) Initialize storage to random data.

- Tests are executed, monitored and controlled by the user from

the BITS Monitor Session. The monitor session screen displays status

continuously during the test. Function keys allow the user to start

testing, enable or disable automatic scrolling of data during the test, enable or disable printing of test status, skip a Wait for Input Unit Message operation, Stop testing, manually scroll up and down to view test status after test operation and access the HELP screens.

SECURITY: Use, copying and distribution of this data is subject to the restrictions in the Agreement For IBM TDB Database and Related Computer Databases. Unpublished - all rights reserved under the Copyright Laws of the

United States. Contains confidential commercial information of IBM exempt

from FOIA disclosure per 5 U.S.C. 552(b)(4) and protected under the Trade

Secrets Act, 18 U.S.C. 1905.

COPYRIGHT STATEMENT: The text of this article is Copyrighted (c) IBM Corporation 1989. All rights reserved.

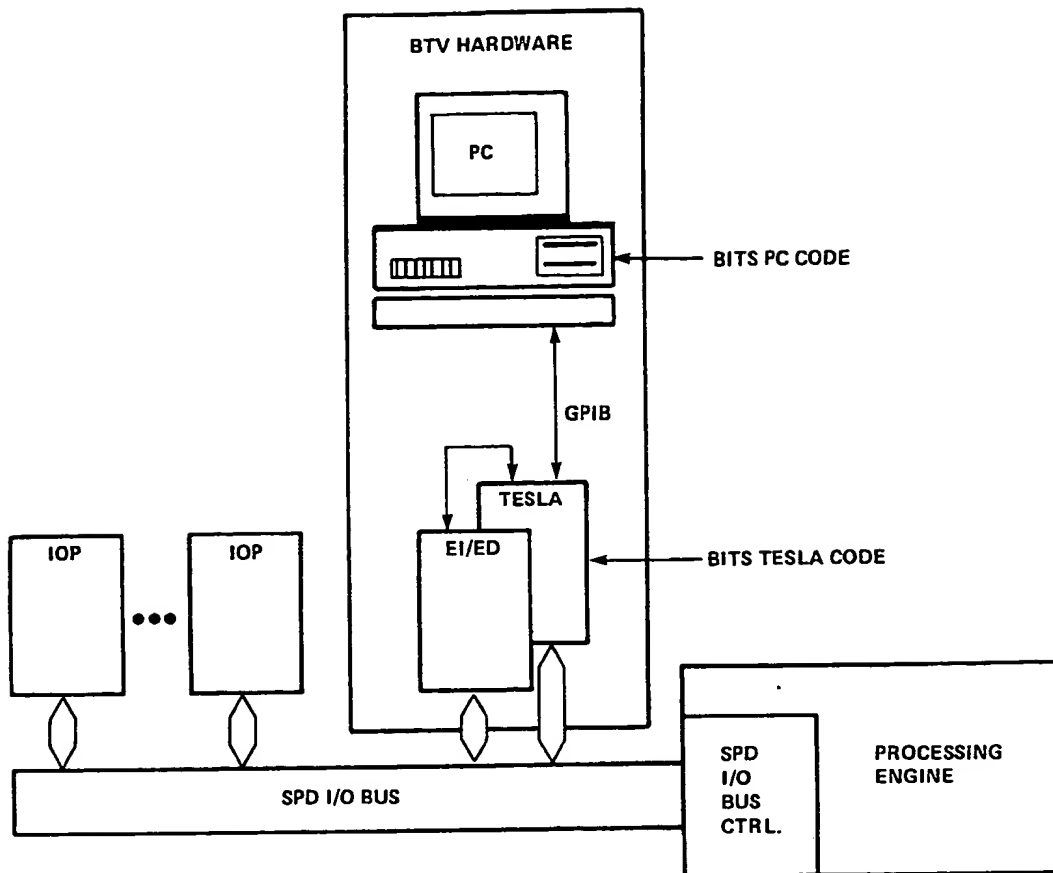


FIG. 1 BITS HARDWARE STRUCTURE

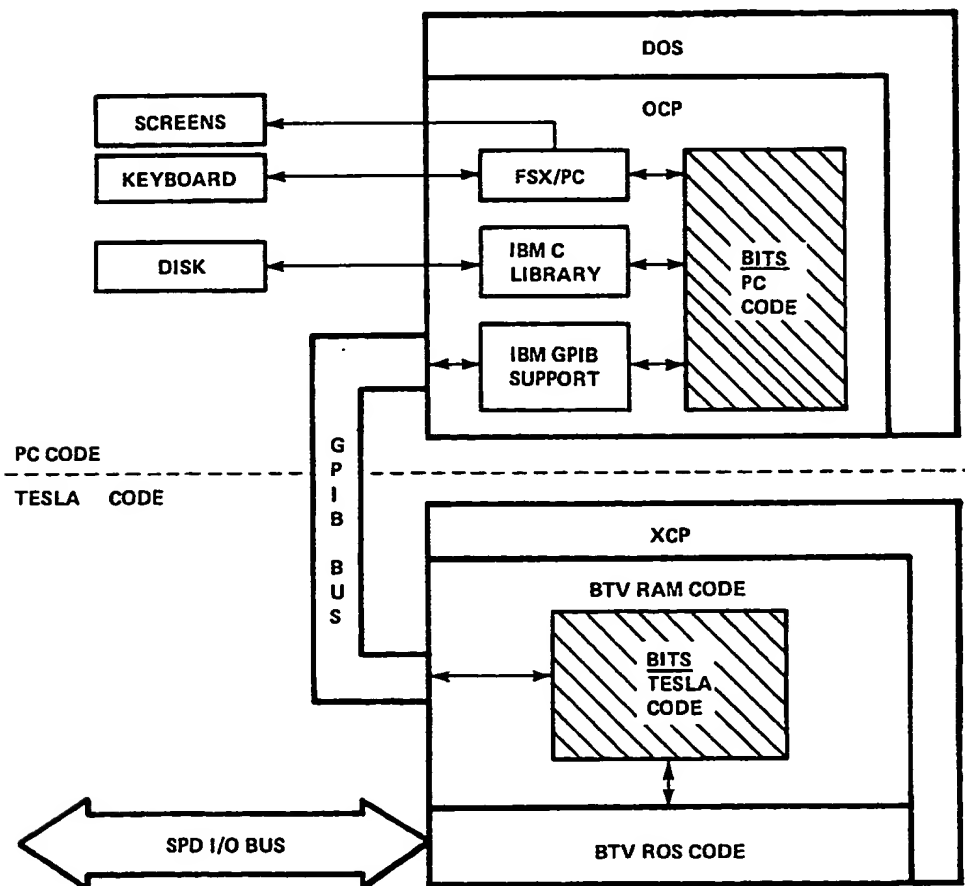


FIG. 2 BITS SOFTWARE STRUCTURE

US-PAT-NO: 6507612
DOCUMENT-IDENTIFIER: US 6507612 B1
TITLE: Bus access controller

----- KWIC -----

Brief Summary Text - BSTX (5):

The above data bus control system has an error detection circuit provided on the first and second buses, and uses the second bus when the error detection circuit detects that the first bus fails. However, bus width for data transfer is selected after a request source starts the data transfer. This creates a problem because the system cannot correct an error on the bus between the request source and the control circuit.

Detailed Description Text - DETX (9):

If device response determination circuit 9 determines that the request is a request to its own device, if REQ64 observing circuit 4 indicates that the request is a 64-bit data transfer request, and if high order bus data parity checker 7 and low order bus data parity checker 8 do not detect a parity error, ACK64 response controller 10 sends an ACK64 signal to the request source. PCI device 1 indicates that PCI device 1 will respond to the 64-bit data transfer, and activates 64-bit data transfer sequencer 11 to start the 64-bit data transfer processing.

Detailed Description Text - DETX (10):

On the other hand, if high order bus data parity checker 7 detects a parity error, ACK64 response controller 10 does not send an ACK64 signal to the request source. PCI device 1 indicates that PCI device 1 will not respond to the 64-bit data transfer, and activates 32-bit data transfer sequencer 12 to start 32-bit data transfer processing. The request source switches the 64-bit data transfer request to the 32-bit data transfer due to the lack of an ACK64 signal. Thus, PCI device 1 continues to execute the data transfer with

the low
order bus.

Detailed Description Text - DETX (14):

In FIG. 2, when a PCI device 1 receives a request address for a 64-bit data transfer, PCI device 1 checks the parity of the data received from a high order bus and a low order bus. In the 64-bit data transfer in a 64-bit PCI bus, effective data of the request address is sent through the low order bus, and the data sent through the high order bus is secured only on parity, but can be treated as invalid data for address information. Therefore, when issuing the request address for a 64-bit data transfer transaction, the high order bus can be regarded as an empty cycle. If PCI device 1 does not detect a parity error in the data received from the high order and low order buses, PCI device 1 outputs an ACK64 signal to the source of the 64-bit data transfer request and activates 64-bit data transfer sequencer 11. 64-bit data transfer sequence 11 starts the 64-bit data transfer transaction.

Detailed Description Text - DETX (15):

On the other hand, when PCI device detects a parity error in the data received from the high order bus, PCI device 1 does not treat this as a critical error, but informs the request source that PCI device 1 cannot perform the 64-bit data transfer.

Detailed Description Text - DETX (22):

Device response determination circuit 9 informs ACK64 response controller 10 that the request is the request to its own device. REQ64 observing circuit 4 informs ACK64 response controller 10 that the request is the 64-bit data transfer request. High order bus data parity checker 7 and low order bus data parity checker 8 inform ACK64 response controller 10 that checkers 7 and 8 do not detect a parity error. ACK64 response controller 10 receives the information, performs the instruction of activating 64-bit data transfer sequencer 11, and responds to the request source with an ACK64 signal to inform the request source of receiving the 64-bit data transfer.

Detailed Description Text - DETX (23):

On the other hand, when high order bus data parity checker 7 detects a parity error, high order bus data parity checker 7 informs ACK64 response controller 10 that the checker 7 detects the parity error. Device response determination circuit 9 informs ACK64 response controller 10 that the request is the request to its own device. REQ64 observing circuit 4 informs ACK64 response controller 10 that the request is the 64-bit data transfer request. Low order bus data parity checker 8 informs ACK64 response controller 10 that the checker 8 does not detect a parity error. ACK64 response controller 10 activates 32-bit data transfer sequencer 12, and informs the request source that PCI device 1 will not receive the 64-bit data transfer, by not responding with an ACK64 signal. Not outputting an ACK64 signal by ACK64 response controller 10, the request source switches the 64-bit data transfer to the 32-bit data transfer. Thus, the PCI device performs data transfer with the low order bus without using the high order bus.

Detailed Description Text - DETX (24):

If a parity error is not detected in the high order bus data and low order bus data of a request address which are sent in period T0, the 64-bit data transfer is performed for one period by using the high order bus and low order bus in period T1 and thereafter. Thus, in period T1, transfer of data D1 and D0 is performed through the high order bus and low order bus, respectively.

Detailed Description Text - DETX (25):

If a parity error is detected in the high order bus data sent during period T0, data D1, which would be transferred through the high order bus in period T1, is transferred through the low order bus in the period T2. Thus, the PCI device performs the 32-bit data transfer through the low order bus in the period T1 and thereafter.

US-PAT-NO: 6216189
DOCUMENT-IDENTIFIER: US 6216189 B1
TITLE: Error master detector

----- KWIC -----

Brief Summary Text - BSTX (3):

The present invention relates to a system including two or more bus masters and a bus arbitrator for receiving requests for bus use of the bus masters to arbitrate the bus use, and more particularly, relates to an error master detector of a bus master for detecting a corresponding bus cycle error generated by the system.

Brief Summary Text - BSTX (7):

Bus errors which occur on a bus where the bus arbitration cycle and the data transfer cycle overlap, are classified into response errors, bus timeout errors and parity errors. Contemporary techniques for detecting bus errors and error recovery are disclosed, for example, U.S. Pat. No. 4,785,453 for High Level Self-Checking Intelligent I/O Controller issued to Chandran et al., U.S. Pat. No. 4,855,234 for Method And Apparatus For Error Recovery In A Multibus Computer System issued to Hartwell et al., U.S. Pat. No. 5,313,627 for Parity Error Detection And Recovery issued to Amini et al., U.S. Pat. No. 5,499,346 for Bus-To-Bus Bridge For A Multiple Bits Information Handling System That Optimizes Data Transfers Between A System Bus And A Peripheral Bus issued to Amini et al., U.S. Pat. No. 5,511,164 for Method And Apparatus For Determining The Source And Nature Of An Error Within A Computer System issued to Brunmeier et al., U.S. Pat. No. 5,537,535 for Multi-CPU System Having Fault Monitoring Facility issued to Maruyama et al., U.S. Pat. No. 5,588,112 for DMA Controller For Memory Scrubbing issued to Dearth et al., and U.S. Pat. No. 5,680,537 for Method And Apparatus For Isolating An Error Within A Computer System That Transfers Data Via An Interface Device issued to Byers et al. Generally, when a bus error occurs, the error type is recorded in a predetermined register, i.e., a status register, and the bus error is

announced
to a processor using an interrupt. In addition, a bus request signal
or a bus
grant signal of a bus master (error master) where an error occurs are
cleared
when the bus cycle begins, and the bus request signal or the bus grant
signal
driven by another bus master is active, i.e., a time difference occurs
between
the bus grant state and the actual bus. As I have observed, however,
only the
fact that the error occurred is announced. The bus cycle of a bus
master where
the error occurred is not announced. Therefore, it is impossible to
rapidly
repair or recover the bus master causing the bus error.

US-PAT-NO: 4750177

DOCUMENT-IDENTIFIER: US 4750177 A
See image for Certificate of Correction

TITLE: Digital data processor apparatus with pipelined
fault tolerant bus protocol

----- KWIC -----

Brief Summary Text - BSTX (47):

A processor module according to the invention detects and locates a fault by a combination of techniques within each functional unit including comparing the operation of duplicated sections of the unit, the use of parity and further error checking and correcting codes, and by monitoring operating parameters such as supply voltages. Each central processing unit in the illustrated computer system, as one specific example, has two redundant processing sections which operate in lock-step synchronism. An error detector compares the operations of the redundant sections and, if the comparison is invalid, isolates the processing unit from transferring information to the bus structure. This isolates other functional units of the processor module from any faulty information which may stem from the processing unit in question. Each processing unit also has a stage for providing virtual memory operation and which is not duplicated. Rather, the processing unit employs parity techniques to detect a fault in this stage.

Brief Summary Text - BSTX (55):

The invention in one embodiment embraces digital data processor apparatus having at least a central processing unit, a random-access memory unit, a control unit for a mass storage device, and a control unit for a communication device, and further featuring a bus structure having redundant first and second buses and a third bus. The buses are connected with all the units for operating the units and for providing information transfers between them. Fault detection means check each information transfer between any unit and any one or more of the first bus and the second bus. The fault detection means

detect fault conditions in a unit and in each of the first and second buses.
The embodiment further features logic means responsive to the fault detection means and responding to the absence of any detected fault condition for providing information transfers on both the first bus and the second bus and responding to the detection of a fault in one of the first and second buses to condition all the units to respond only to information-transferring signals on the other of the first and second buses.

Detailed Description Text - DETX (92):

Further, each unit applies address and data signals on the A and B buses with parity which that unit generates. The memory unit serves, in the illustrated embodiment, to check bus parity and to drive the appropriate bus error line of the X bus 46 during the timing interval immediately following the interval in which it detected the bus parity error. The memory unit also sets a diagnostic flag and requests a diagnostic interrupt.

Detailed Description Text - DETX (121):

Any mismatch of corresponding signals applied to the comparator 12f causes a comparison error signal, which is applied to a common, non-duplicated control stage 86. In response, the control stage sends out error signals on the X bus 46. It also disables the drivers in the transceivers 12e to take the processing unit 12 off-line so that it cannot send further signals to other units of the FIG. 1 system. The control stage 86 also monitors the two parity error signals from the parity check circuits 82 and 84. The control stage 86 is part of the CPU control section 12d (FIG. 1), which also includes clamp circuits 88 and 90. The clamp circuits respond to a power failure at the unit 12 to clamp to ground, at the transceiver 12e drivers, all output lines from the processing unit 12 to the bus structure 30.

Detailed Description Text - DETX (148):

The detection of an error by the parity checking circuits 82 and 84 connected with the virtual memory MAP 80 produces a parity error signal which also is applied to the control stage 133.

Detailed Description Text - DETX (179):

The illustrated memory unit 16 also has a status and control stage 368 which is not duplicated. The stage receives the parity error signals, the comparator fault signals, and ECC syndrome signals from the ECC stage 320. The stage 368 connects to numerous other elements in the memory unit, with connections which are in large part omitted for clarity of illustration. A bus error stage 370 is connected with the stage 368 and, by way a transceiver, with conductors of the X bus 46 as described below with reference to FIG. 10.

Detailed Description Text - DETX (181):

FIG. 10 shows the bus error stage 370 of FIG. 9 which responds to parity error signals and the ECC syndrome signal of the illustrated memory unit 26. An OR gate 372 receives the Data Parity Error signal for the A bus, which the parity check circuit 328 produces on its output line 328a, and receives the Address Parity Error signal for the A bus output from the parity check circuit 364 on line 364a. Similarly, the Data Parity Error signal for the B bus, produced on line 330a, and the Address Parity Error signal for the B bus, produced on line 366a, are applied to a further OR gate 374. Either error signal for the A bus and input to the OR gate 372 actuates a transceiver 376 to produce an A Bus Error signal. This signal is applied to the X bus 46 for communication to all units in the module 10. Similarly, an error signal for the B bus and input to the OR gate 374 actuates a further transceiver 378 to produce a B Bus Error signal that is applied to the X bus 46. FIG. 2 illustrates operation of the illustrated processor module 10 when either Bus Error signal is asserted.

PAT-NO: JP403255744A
DOCUMENT-IDENTIFIER: JP 03255744 A
TITLE: ERROR DETECTING SYSTEM
PUBN-DATE: November 14, 1991

INVENTOR-INFORMATION:
NAME
NISHITO, KATSUHIKO

ASSIGNEE-INFORMATION:
NAME COUNTRY
NEC CORP N/A

APPL-NO: JP02054161
APPL-DATE: March 6, 1990

INT-CL (IPC): H04L012/40, H04L012/28
US-CL-CURRENT: 714/800

ABSTRACT:

PURPOSE: To detect an error by adding a parity bit to data to be outputted to a bus for a master, detecting the parity error of the data outputted from the master to the bus for each slave and notifying a detected result to the master by a wired OR.

CONSTITUTION: A parity bit adding means provided on the master 1 adds the parity bit to the data to be outputted to a bus 3. Parity error detecting means (26-1)-(26-n) provided on respective slaves (2-1)-(2-n) detect the parity error of the data outputted from the master 1 to the bus 3. The detected results are notified from the parity error detecting means (26-1)-(26-n), which are provided on the respective slaves (2-1)-(2-n), to the master 1 by the wired OR.

COPYRIGHT: (C)1991,JPO&Japio